A fast top–down method for constructing reliable radiation hybrid frameworks

Trond H. Bø1,∗, Inge Jonassen1, Ingvar Eidhammer1 and Carsten Helgesen2

1Department of Informatics, University of Bergen, HIB, N5020 Bergen, Norway and
2Cap Gemini Norway, Vestre Strømkaien 5, 5008 Bergen, Norway

ABSTRACT

Motivation: Radiation Hybrid Mapping (RHM) is a technique used to order a set of markers on a genome and estimating physical distances between them. RHM provides information on marker placement independent from other methods such as sequencing, and can therefore be used for example in genome sequencing to help ordering contigs. A radiation hybrid framework can be constructed by choosing a set of markers so that the chromosome coverage is good and so that the markers can be ordered with high confidence. Automatically constructing RHM frameworks is a computationally challenging problem.

Results: We have developed a new method for constructing radiation hybrid frameworks. Given a relatively large set of markers for a chromosome, the algorithm aims to select an ordered subset that makes up a framework, and that contains as many markers as possible. The algorithm has a time complexity that is better than any of the existing methods that we are aware of. Furthermore, we propose a method for comparing if two frameworks are consistent, giving a visual presentation as well as quantitative measures of how well the two frameworks agree. Applying our method on marker sets from 22 human chromosomes and comparing the resulting frameworks with previously published frameworks, we demonstrate that our automatic method efficiently constructs frameworks with good coverage of each chromosome and with high degree of agreement on the marker ordering.

Availability: The software is available from the authors.

Supplementary information: MCST paths, frameworks and plots are available at http://www.ii.uib.no/~trondb/RHmapping/

Contact: trondb@ii.uib.no

INTRODUCTION

Radiation Hybrid Mapping (RHM) is a technique that can be used to estimate the order of a set markers along a genome. The markers are typically Sequence-Tagged Sites (STSs) or Expressed Sequence Tags (ESTs). In a shotgun sequencing project RH maps provide information that can be used for ordering clones and contigs and in the gap closure phase. Furthermore RH maps can be used to analyze and assess intermediate and final results of genome sequencing projects. For example, Olivier et al. (2001) used RH maps to analyze two different human genomic sequence assemblies. For organisms whose genomes will not be completely sequenced, at least not in the near future, RH maps can be of great value. In order to make full use of the RHM techniques, automatic methods are needed that can construct reliable ordering of STSs given RHM data.

Each marker is scored against an ordered set of hybrid cells, normally referred to as a panel, containing DNA fragments from the organism to be mapped. The result is a score vector of 0s, 1s and 2s. A 0 in position i indicates that the hybrid cell i in the panel does not contain any fragment with the marker sequence, a 1 indicates that it does contain a fragment with the marker sequence, and a 2 indicates that the score result for some reason is unknown.

Problem definition

Boehnke and co-workers (Boehnke et al., 1991; Boehnke, 1992; Lange and Boehnke, 1992) have developed methods to evaluate the orders in relation to the RH data used to produce them. The methods each return a number quantifying the result of the evaluation. However the numbers cannot be directly interpreted as a measurement of the map quality, but can be used to compare different orderings to find the one that is best supported by the RH data and therefore most likely to be biologically correct. Ordering the markers thus becomes the problem of finding an order where no alternative orders are considered ‘better’. For n markers there are $\frac{n!}{2}$ orders (reversed orders not counted), so finding an optimal order is a computationally demanding, if not impossible task. In practice the markers are ordered by searching though a subspace of all orders, choosing the order that is scored best by one of the two objective functions Obligate

∗To whom correspondence should be addressed.
Chromosome Breaks (OCB) or multipoint Maximum Likelihood (ML) (see Section Methods for explanation).

The number of markers for each human chromosome has currently reached hundreds, and for some even thousands, making it virtually impossible to find the best order for two reasons. First, the number of hybrid cells used in the experiments is limited and consequently the resulting RH data may not provide enough information to define one best order for all the markers (e.g., many markers may have identical score vectors). Secondly as the number of possible orders becomes astronomical, the algorithmic problem of searching for the best order becomes computationally very hard. Furthermore even if the best scoring order has been found, there may be several other orders that receive only a marginally worse score. One method for dealing with this is framework construction where a subset of the markers is chosen for which an order is found that is significantly better than all other orders of the chosen markers.

This means that given a set of markers \( X = \{ x_1, \ldots, x_n \} \), a subset \( Y \) of \( X \) is chosen and the markers in \( Y \) are ordered. The identified ordering of the markers in \( Y \) is required to be significantly better scoring than the second best known order of the markers in \( Y \). One formal definition is that a framework is required to be 1000 times more likely than the second best ordering of the framework markers as scored by the ML function. Deloukas et al. (1998) use a 2.5 ML score difference to the second best scoring order, meaning their framework is at least 316 times more likely than the second best order (101.55 \( \approx \) 316). Each of the markers not included in the framework (e.g., the markers in \( X - Y \)) can be placed relative to the framework by trying all intervals, and choosing the one(s) giving the highest ML score. These markers are thus not ordered, only assigned to the most likely interval(s) relative to the framework.

The focus of this paper is on the framework construction problem: given a (large) unordered set of markers \( X \), find a subset \( Y \) of \( X \) such that the markers in \( Y \) can be ordered to qualify a 1000:1 framework criteria, and find this order for \( Y \).

**Summary of previous methods**

*Marker ordering.* Several software packages exist for finding a close to optimal order for a set of markers. Packages like RHMAP by Boehnke et al. (1996) and Radiation Hybrid Ordering (RHO) by Ben-Dor et al. (2000) use optimization algorithms to search for optimal orders for a set of markers. Agarwala et al. (2000) used an optimization package called CONCORDE to order a set of markers. Both Ben-Dor et al. and Agarwala et al. reduce the marker ordering problem to the well known and thoroughly studied Traveling Salesman Problem (TSP), and then apply optimization algorithms known to perform well on the TSP on the reduced problem and finally ‘translates back’ the TSP solution to a marker ordering. Ben-Dor et al. use the Simulated Annealing heuristic and Agarwala et al. use the Lin–Kernighan heuristic. Both these groups have demonstrated that their approach is successful in producing marker orders of high quality.

*Framework construction.* Some software packages (RHMAPPER, Stein et al., 1995; Slonim et al., 1997; Soderlund et al., 1998; Multimap, Matise et al., 1994) have built in methods for building frameworks based on radiation hybrid data, assembling the map in a bottom up fashion. In RHMAPPER 1000:1 triples, meaning 1000:1 frameworks of 3 markers, are used. Combining overlaps between such triples, larger framework orders can be constructed. Overlaps can be on the forms (1) a-b-c and b-c-d or (2) a-b-c and a-c-d, both producing a-b-c-d. RHMAPPER has one algorithm constructing frameworks only from overlaps of type (1), and another constructing frameworks from overlaps of type (1) and (2). The time complexity for producing a map with RHMAPPER is \( O(n^3) \) for an input set of \( n \) markers (see Section Time complexity). In Multimap another approach is used, constructing a framework one marker at each step, starting with two markers in the first step. In Section Time complexity we demonstrate that also this method uses at least \( O(n^3) \) time.

Agarwala et al. (2000) try building framework maps by first selecting a subset of markers that are ‘evenly spaced’, meaning that no two markers in the set are closer to each other than a specified threshold. Then they use five different reduction variants to reduce the ordering problem into the TSP. By applying the Lin–Kernighan heuristic on each instance they obtain five orderings of the same markers. Then they use the largest subsequence common to the five orders as the framework order. However, they do not make any attempt to compare the identified ordering to the second best ordering and consequently they cannot guarantee that the orders are 1000:1 frameworks.

*Marker ordering versus framework construction.* A fundamental difference between marker ordering algorithms and framework construction algorithms is that a marker ordering algorithm cannot guarantee that the identified ordering is significantly better than the second best (there can be multiple approximately equally good orders), while framework construction algorithms are designed to give such a guarantee. A marker ordering algorithm searches through a subspace of all orders including all the input markers, reporting the best order(s) found during the search. It is not possible to find (e.g. 1000:1) framework orders for all sets of input markers. In order for a marker ordering algorithm to find a framework order it therefore depends on getting as input a set of markers for which a framework order can be found—we call such a marker
set a framework set. Additionally, the ordering algorithm must be able to find the best order of a framework set. The problem is therefore to find such a framework set, and be able to order the markers correctly. Framework construction algorithms are designed to deal with this problem, carefully choosing a set of markers that can be ordered to satisfy the required properties of a framework.

Overview of the current method

The method described in this paper uses a novel approach to framework construction, first making a high scoring seed order and then finding a framework as a subsequence order. Our seed orders are made by first constructing a Maximum Cost Spanning Tree (MCST) in a graph where there is a node for each input marker, and each edge has as weight the 2-point LOD score (see Cox et al., 1990, for an explanation). Then the longest path (containing most markers) in the MCST is found and used as the seed order. By then seeking through the seed order, we find the largest subsequence order of the seed order that consists of overlapping 1000:1 triples, by use of dynamic programming. Producing a framework with this method can be done in $O(n^2)$ time, as demonstrated in Section Time complexity, a factor of $n$ faster than RHMAPPER and Multimap.

METHODS

Scoring functions

Scoring functions are used to rank orders relative to each other, such that if we have two orders $O_1$ and $O_2$ for the same set of markers $X$, we can conclude that one of them is more likely to be the true biological order than the other. If $O_1$ scores ‘better’ than $O_2$, then $O_1$ is assumed to be closest to the true order.

There are two such scoring functions in use, called OCB and multipoint ML. OCB is based on counting transitions between 0 and 1 in the hybrid vectors given by the ordered score vectors. Thus if an order gives the hybrid vector $(1, 1, 0, 2, 0, 1, 2, 0)$ this hybrid contributes with three breaks, one between position two and three, one between five and six, and one between six and eight. Note that the 2s are overlooked when counting transitions. The best order scored by OCB is the order minimizing the number of transitions counted over all hybrids. A more detailed discussion of OCB is found in Boehnke (1992).

ML is a statistical method based on Markov models and Bayesian methods, giving the ML to obtain the score vector data given the order we want to test. A detailed discussion of this method can be found in Lange and Boehnke (1992) and Boehnke et al. (1991). A ML score is given as the log$_{10}$ ratio of the likelihood. The higher score an order gets by ML the better it is considered to be.

Making a seed order

A seed order should be of high quality and cover as much of a chromosome as possible. It does not need to contain all markers for a chromosome, as long as all parts of the chromosome are represented by at least some markers. Finding close to optimal orders is not a trivial task, since the number of possible orders explodes as the number of input markers increases. We have used the following graph based heuristic, finding a high quality order for a relatively large subset of the markers.

Given a set of markers $X = \{x_1, \ldots, x_n\}$, we construct the complete graph $G = (V, E)$ such that $V = X$ and the weight on each edge $e(x_i, x_j) = e_{i,j}$ is the 2-point LOD score between the markers. A MCST $T$ is then constructed for $G$ by a greedy algorithm (Manber, 1989, p. 210), and the path $P$ in $T$ containing the largest number of markers is found by Depth-First Search (DFS; Manber, 1989, p. 190) starting from each leaf in $T$. Given the set $Y$ containing the markers in $P$, $P$ will be a MCST for the complete graph for the markers in $Y$. Since $P$ is a MCST for the markers in $Y$, the order given by $P$ maximizes the sum of LOD scores, counted between neighbor markers. Based on this we assume that the order given by $P$ is a suitable candidate for framework searching.

Searching in the seed order

To search through a seed order $O = (o_1, o_2, o_3, \ldots, o_n)$ for a framework order, we consider ordered triples $(o_i, o_j, o_k)$, $0 < i < j < k < n + 1$, in $O$ that are 1000:1 frameworks, also called 1000:1 triples. A triple $(o_i, o_j, o_k)$ is 1000:1 if it is at least 1000 times more likely than the alternative orderings of $o_i$, $o_j$ and $o_k$. A search for a framework order in $O$ is done by finding one largest subsequence order $O' = (o_{i_1}, o_{i_2}, \ldots, o_{i_m})$, $t_1 < t_{i+1}$, where every triple $(o_{i_1}, o_{i_{1+1}}, o_{i_{1+2}})$, $0 < i < m$, is a 1000:1 triple. Note that there can be several equally large $O'$s in $O$.

Considering any 1000:1 triple in $O$, $(o_{i_1}, o_{i_{1+1}}, o_{i_{1+2}})$, a largest framework order starting with this specific triple can easily be found if a largest framework order starting with $o_{i_1}$ and $o_{i_2}$ is known. This information can be stored in a 2-dimensional $n \times n$ table $L$ that has two elements largest and next in each cell. The entry $L(i, j).\text{largest}$ gives the number of triples in a largest framework order of overlapping 1000:1 triples starting with $o_i$ and $o_j$, and $L(i, j).\text{next}$ gives the next marker in the order. All entries $L(i, n).\text{largest}$ can be initialized with the value 0, as there are no markers after $o_n$ in the seed order. Assuming the entries $L(j, k)$ are already computed for all $k > j$, the values in $L(i, j)$ are determined from $L(j, k)$ and where $(o_i, o_j, o_k)$ is a 1000:1 triple. This is achieved by calculating row $n$ in $L$ first, then row $n - 1$, and so on. Defining $K_{i,j}$ and $K^*_i,j$ as $K_{i,j} = [k : (o_i, o_j, o_k)$ is a 1000:1 triple] and $K^*_i,j =
Fig. 1. The find_frw algorithm in pseudo code. First the length of the largest framework starting with $o_i$ and $o_j$ is computed for all $i$ and $j$, $1 \leq i < j \leq n$. Then the largest framework is backtracked in the table $L$ by the next values, starting with the markers $o_i$ and $o_j$ such that $L(i,j)$.largest has the highest largest value in $L$.

```plaintext
find_frw(0)
{
    n=0;
    /*Initialize L*/
    For i=1 to n
    L(i,n).largest=0

    /*Compute all entries in L*/
    For i=n-2 down to 1
    { For j=i+1 to n-1 {
        For k=1 to n {
            If ((i,j,k) is a 1000:1 triple) {
                If (L(j,k).largest+1) > L(i,j).largest) {
                    L(i,j).largest=L(j,k).largest+1
                    L(i,j).next=k
                    L(i,j).weakest_link=min(odd((i,j,k)), L(i,j).weakest_link)
                }
                If (L(i,k).largest+1) > L(i,j).largest) {
                    L(i,j).weakest_link>max(odd((i,j,k), L(i,j).weakest_link) {
                    L(i,j).next=k
                    L(i,j).weakest_link=min(odd((i,j,k)), L(i,j).weakest_link)
                }
            }
        }
    }
    /*Backtrace the framework order*/
    Find some values i and j such that L(i,j).largest is largest in L (there may be several pairs (i,j)).
    $O'(1)=0(i)$
    $O'(2)=0(j)$
    t=0
    While (L(i,j).largest > 0) {
        k=L(i,j).next
        t=t+1
        $O'(t)=0(k)$
        i=j
        j=k
    }
    Return the order $O'$. 
}
```

where odds$(o_i,o_j,o_k)$ is the difference in likelihood, computed with ML, between $(o_i,o_j,o_k)$ and the second best ordering of $o_i$, $o_j$ and $o_k$ (note that the reversed order is considered the same order). We then choose the $k^* \in K^*_{i,j}$ for which $L(j,k^*).weakest_link$ get the highest value, replacing $k^*$ with $k^{**}$ in formula (1) and (2). When all the cells in $L$ are computed, a largest framework $O'$ of 1000:1 triples can be obtained by tracing the next values, starting at a cell $L(i,j)$ with the highest largest value. The algorithm for searching in a seed order, find_frw is given in pseudo code in Figure 1.

Fig. 1. As there are $O(n^3)$ triples in an order containing $n$ markers, and all triples have to be examined to compute the values of all cells in $L$, the find_frw algorithm has a time complexity of $O(n^3)$. This can be improved by only computing a band above
Multimap builds a framework of \( k \) markers in \( k \) steps. At each step \( i, i = 2, 3, \ldots, k \), at least one marker is tested against all intervals in the framework. Each step takes at least \( O(i^2) \) time: \( i \) intervals times \( O(i) \) to compute the ML score of an order of length \( i \). In step \( k \) no more markers can be placed on the framework map, which means that all the remaining \( n - k \) markers are tested against all intervals in a framework of size \( k \). For each of the \( n - k \) markers, \( k \) intervals are tested taking \( O(k) \) time for each test. This gives a total time complexity for step \( k \) of \( O((n - k)k^2) \), and, for example if \( k \approx \frac{n}{2} \), the time complexity for step \( k \) become \( O(n^3) \).

**RESULTS**

**Datasets**

For testing purposes we used the GB4 human data from RHdb\(^1\) (Rodriguez-Tomé and Lijnzaad, 2000) release 17.0. To be able to compare our maps with the GeneMap98 maps, eight of the scores in the score vectors for each marker were removed from the RHdb data, because the GeneMap98 maps do not use these hybrids. Two additional filters were applied to the data. The first filter removes from the dataset all markers with more than three 2s in the score vector, 2 meaning uncertain scores. This is done to reduce the influence of lacking data on the results. The second filter removes redundant markers by reducing all markers having indistinguishable score vectors to be represented by only one marker, preferring the marker having the fewest 2s in it’s score vector.

The markers were given as input in the order which they are given in RHdb, meaning that they were ordered by their RHdb entry number in the input file. This order has no relation to the GeneMap98 order.

**Evaluation of the approach**

The two steps for framework construction have been implemented in a program able to analyze hundreds of markers in a few minutes on a 300 MHz Sun Ultra 10 workstation, and has been successful in producing high quality frameworks of considerable size as output.

To test the reliability of our approach, we tested it on a dataset of markers for each of the human chromosomes 1–22, making one tentative framework for each. The results are summarized in Table 1.

Each output framework order was compared to the corresponding GeneMap98 chromosome framework\(^2\). The comparison was carried out by plotting each marker in the predicted framework order relative to the GeneMap98 framework. This was done by binning each of

---

\(^1\)http://www.ebi.ac.uk/RHdb/index.html.

\(^2\)The GeneMap98 frameworks for the human chromosomes are publicly available at: http://www.sanger.ac.uk/Software/RHserver/RHserver.shtml. These frameworks were published by Deloukas *et al.* (1998).
Table 1. An overview of the tests done, giving an indication of the quality of our frameworks compared with the GeneMap98 frameworks. An explanation of interval coverage, agreement vs GeneMap98 framework, and quality is given in the text. The last column gives the runtime on a 300 MHz Sun Ultra 10 workstation. MCST paths, frameworks and plots are available at: http://www.ii.uib.no/~trondb/RHmapping/

<table>
<thead>
<tr>
<th>Chromosome No.</th>
<th>No. of input markers</th>
<th>No. of markers in GM98 frw</th>
<th>No. of markers in longest MCST path</th>
<th>No. of markers in our frw</th>
<th>Agreement versus GeneMap98</th>
<th>Interval coverage</th>
<th>Quality</th>
<th>Reversed sections</th>
<th>Runtime (min : s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3020</td>
<td>83</td>
<td>379</td>
<td>91</td>
<td>1</td>
<td>0.79</td>
<td>0.79</td>
<td>N</td>
<td>16 : 04</td>
</tr>
<tr>
<td>2</td>
<td>2212</td>
<td>88</td>
<td>320</td>
<td>74</td>
<td>1</td>
<td>0.74</td>
<td>0.74</td>
<td>N</td>
<td>9 : 13</td>
</tr>
<tr>
<td>3</td>
<td>1903</td>
<td>82</td>
<td>253</td>
<td>67</td>
<td>0.73</td>
<td>0.45</td>
<td>0.33</td>
<td>Y</td>
<td>6 : 20</td>
</tr>
<tr>
<td>4</td>
<td>1457</td>
<td>58</td>
<td>210</td>
<td>40</td>
<td>0.57</td>
<td>0.37</td>
<td>0.21</td>
<td>Y</td>
<td>4 : 01</td>
</tr>
<tr>
<td>5</td>
<td>1412</td>
<td>63</td>
<td>212</td>
<td>60</td>
<td>1</td>
<td>0.81</td>
<td>0.81</td>
<td>N</td>
<td>4 : 23</td>
</tr>
<tr>
<td>6</td>
<td>1959</td>
<td>66</td>
<td>308</td>
<td>90</td>
<td>1</td>
<td>0.81</td>
<td>0.81</td>
<td>N</td>
<td>8 : 22</td>
</tr>
<tr>
<td>7</td>
<td>1388</td>
<td>64</td>
<td>235</td>
<td>67</td>
<td>0.99</td>
<td>0.71</td>
<td>0.70</td>
<td>N</td>
<td>2 : 33</td>
</tr>
<tr>
<td>8</td>
<td>1162</td>
<td>41</td>
<td>185</td>
<td>55</td>
<td>1</td>
<td>0.88</td>
<td>0.88</td>
<td>N</td>
<td>2 : 52</td>
</tr>
<tr>
<td>9</td>
<td>1014</td>
<td>45</td>
<td>161</td>
<td>43</td>
<td>1</td>
<td>0.76</td>
<td>0.76</td>
<td>N</td>
<td>1 : 58</td>
</tr>
<tr>
<td>10</td>
<td>1300</td>
<td>61</td>
<td>230</td>
<td>61</td>
<td>1</td>
<td>0.82</td>
<td>0.82</td>
<td>N</td>
<td>4 : 26</td>
</tr>
<tr>
<td>11</td>
<td>1389</td>
<td>50</td>
<td>181</td>
<td>48</td>
<td>0.60</td>
<td>0.35</td>
<td>0.21</td>
<td>Y</td>
<td>2 : 53</td>
</tr>
<tr>
<td>12</td>
<td>1425</td>
<td>44</td>
<td>197</td>
<td>57</td>
<td>0.75</td>
<td>0.62</td>
<td>0.47</td>
<td>Y</td>
<td>3 : 55</td>
</tr>
<tr>
<td>13</td>
<td>682</td>
<td>41</td>
<td>116</td>
<td>32</td>
<td>1</td>
<td>0.67</td>
<td>0.67</td>
<td>N</td>
<td>0 : 58</td>
</tr>
<tr>
<td>14</td>
<td>954</td>
<td>37</td>
<td>171</td>
<td>34</td>
<td>1</td>
<td>0.63</td>
<td>0.63</td>
<td>N</td>
<td>2 : 25</td>
</tr>
<tr>
<td>15</td>
<td>981</td>
<td>34</td>
<td>155</td>
<td>43</td>
<td>1</td>
<td>0.74</td>
<td>0.74</td>
<td>N</td>
<td>2 : 01</td>
</tr>
<tr>
<td>16</td>
<td>971</td>
<td>40</td>
<td>91</td>
<td>30</td>
<td>0.57</td>
<td>0.15</td>
<td>0.08</td>
<td>Y</td>
<td>0 : 52</td>
</tr>
<tr>
<td>17</td>
<td>1213</td>
<td>33</td>
<td>183</td>
<td>37</td>
<td>1</td>
<td>0.50</td>
<td>0.50</td>
<td>N</td>
<td>2 : 34</td>
</tr>
<tr>
<td>18</td>
<td>640</td>
<td>32</td>
<td>140</td>
<td>36</td>
<td>0.97</td>
<td>0.76</td>
<td>0.74</td>
<td>N</td>
<td>1 : 38</td>
</tr>
<tr>
<td>19</td>
<td>921</td>
<td>26</td>
<td>127</td>
<td>30</td>
<td>0.47</td>
<td>0.48</td>
<td>0.22</td>
<td>Y</td>
<td>1 : 35</td>
</tr>
<tr>
<td>20</td>
<td>782</td>
<td>35</td>
<td>144</td>
<td>34</td>
<td>0.97</td>
<td>0.67</td>
<td>0.65</td>
<td>N</td>
<td>1 : 52</td>
</tr>
<tr>
<td>21</td>
<td>359</td>
<td>19</td>
<td>72</td>
<td>23</td>
<td>1</td>
<td>0.80</td>
<td>0.80</td>
<td>N</td>
<td>0 : 24</td>
</tr>
<tr>
<td>22</td>
<td>493</td>
<td>15</td>
<td>88</td>
<td>22</td>
<td>0.95</td>
<td>0.81</td>
<td>0.78</td>
<td>N</td>
<td>0 : 36</td>
</tr>
</tbody>
</table>

our framework markers to the most likely interval on the GeneMap98 framework, interval meaning placement between two GeneMap98 markers or on the ends of the framework. The binning was achieved by computing the ML score of each possible order consisting of the GeneMap98 framework markers (in framework order) plus one additional marker. The highest scoring order thus decides which interval the additional marker most likely belongs to. For each marker of our framework the highest scoring interval using ML on the GeneMap98 framework is plotted with a ‘*’, while other intervals scoring between 0 and up to 10 times less likely than the highest scoring interval are plotted with a ‘.’. This gives a visualization of how well the framework agree with and cover the GeneMap98 framework. Examples of plots are given in Figure 3.

To give a quantitative measurement of map quality, we define two measures reflecting the quality of a map relative to the GeneMap98 framework. The first, called agreement, considers how many markers of the framework can be given the same order by the GeneMap98 framework. Given a framework order $F = (f_1, f_2, \ldots, f_k)$, the agreement with the GeneMap98 framework is determined by the number of markers in the order $F' = (f_{t_1}, f_{t_2}, \ldots, f_{t_m})$, $1 \leq t_1 < t_2 < \cdots < t_m \leq k$, such that $F'$ is a largest subsequence order of $F$ that ‘agrees’ with the GeneMap98 framework. That $F'$ ‘agrees’ with the GeneMap98 framework means that the order in $F'$ is the same as the order given by mapping each marker on the GeneMap98 framework, allowing to choose one of the most likely intervals for each marker (up to 10 times less likely than best)—those plotted with ‘*’ or ‘.’). The result is given as the ratio $m/k$.

The second quality measure, called coverage, reflects how well the order $F'$ covers the GeneMap98 framework. Given that there are $n$ intervals to which a marker can be assigned on the GeneMap98 framework, the coverage is given as the ratio $j/n$, where $j$ is the number of intervals where a marker from $F'$ is placed, taking the most likely interval for each marker using ML.

Multiplying coverage with agreement gives an overall quality measure of the framework, considering both how well the map agrees with and how well it covers the GeneMap98 framework. Scores will be in the range $[0, 1]$, where 1 is best.

The results obtained for human chromosomes 1–22 are given in Table 1. It shows that in 16 of the 22 cases the frameworks constructed agrees 95% or better with the GeneMap98 frameworks. The lower ratio of the remaining 6 is a result of reversed parts as illustrated in Figure 3b.
Top–down method for RHM framework construction

Fig. 3. Example of plots of a constructed framework against GeneMap98 framework. The order of the constructed framework is along the x-axis, while intervals in the GeneMap98 framework is along the y-axis. Note that in plot (a), some markers are given two ‘*’ intervals. This indicates that the score vector for the marker is equal to the score vector of the GeneMap98 framework marker between these two intervals. The two examples are the plots for the constructed frameworks of chromosome 9 (a) and 11 (b). Plot (b) illustrates the problem of orienting parts relative to each other which is discussed in Section Discussion.

The coverage of the produced frameworks shows more variation. This is partly due to the way the coverage measure is defined penalizing an order if it does not have markers within every single interval in the GeneMap98 framework. This means that even frameworks spanning the entire GeneMap98 framework (like the framework for chromosome 9 shown in Figure 3a) need not get a score of 1.

The MCST paths used as seed orders contained between 9.4 and 21.9% of the input markers, with an average longest path of 15.7% of the input markers. The average for the paths giving frameworks with reversed parts was 13% while the average for paths giving frameworks not having reversed parts was 16.8%. Of the seven paths with lowest percentage, six of them were the paths giving reversed sections.

DISCUSSION

We have demonstrated an approach capable of analyzing hundreds of markers in a matter of minutes, giving relatively large and reliable frameworks as output. The first step in this approach, making a seed order, can in practice be done by other ordering algorithms capable of making high quality orders, but a graph based method using a MCST was found to be efficient in terms of time usage and effective in producing orders of high quality.

The second step searches for a framework that is already inside the seed order, using 1000:1 triples and dynamic programming. Thus the framework found depends totally on the quality of the seed order, as the framework markers have to be ordered correctly in the seed order to be found. A seed order of bad or medium quality will therefore not give as good results, in terms of a large framework, as a better order.

Furthermore we have proposed a way to evaluate the constructed frameworks by comparing them to trusted frameworks. The results of such a comparison can be represented graphically or be summarized in terms of two quality measures. The proposed framework construction method has been tested by applying it to datasets for 22 human chromosomes and comparing the resulting frameworks to GeneMap98 frameworks.

One problem that has occurred is wrong orientation of parts of the framework relative to the GeneMap98 framework. An example is given in Figure 3b. In the cases we have studied the two parts relate to the two chromosome arms, where the arms have not been correctly oriented relative to each other. It seems that good frameworks have been found for each part/arm, but that the full map makes little sense as the relative orientation of the parts are incorrect in the larger framework. Based on score vector data only, it seems difficult to overcome this problem. Given some additional information, such as labeling some markers by their chromosomal location, it could be possible to avoid the problem.

Further work will include using alternative marker ordering methods to produce seed orders. The results of the alternative methods can be assessed in terms of how good frameworks can be constructed. In this work we also plan to explore modifications of the MCST heuristic marker ordering method proposed in this paper. In addition, other filters will be added to refine the sets of markers used as input. For these comparative studies

17
a more comprehensive set of test data would be valuable, and we plan to include RH data from other organisms, e.g. rat and mouse.

ACKNOWLEDGEMENTS
I.J. has been supported by grants from the Norwegian Research Council. The authors thanks three anonymous reviewers for their helpful comments.

REFERENCES